

Nlist

Vulnerable to TOCTOU issues

Sean Barnum, Cigital, Inc. [vita¹]

Copyright © 2007 Cigital, Inc.

2007-04-02

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 6807 bytes

Attack Category	<ul style="list-style-type: none">• Path spoofing or confusion problem• Privilege Exploitation	
Vulnerability Category	<ul style="list-style-type: none">• Indeterminate File/Path• TOCTOU - Time of Check, Time of Use	
Software Context	<ul style="list-style-type: none">• File Management	
Location	<ul style="list-style-type: none">• nlist.h	
Description	<p>Privileged processes calling nlist() should beware of the possibility of an unexpected file being substituted as the operand.</p> <p>The nlist() function returns symbol table information for the specified symbol names, for the executable file whose name is supplied as an argument.</p> <p>Use of nlist() may be subject to attack if a check to ensure that the right file is going to be examined is followed by a use of nlist(); an attacker could conceivably change what file the name refers to in between the check and the use. Depending on how the results returned by nlist() are going to be used, this could be a problem.</p>	
APIs	Function Name	Comments
	nlist	
Method of Attack	<p>The key issue with respect to TOCTOU vulnerabilities is that programs make assumptions about atomicity of actions. It is assumed that checking the state or identity of a targeted resource followed by an action on that resource is all one action. In reality, there is a period of time between the check and the use that allows either an attacker to intentionally or another interleaved process or thread to unintentionally change the state of the targeted resource and yield unexpected and undesired results.</p> <p>An attacker could potentially convince a privileged program to examine the symbol table of an executable to which the attacker would not normally</p>	

1. <http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html> (Barnum, Sean)

	have direct access. If the attacker can gain access to this symbol table information, this could help the attacker to formulate an attack against that executable.		
Exception Criteria			
Solutions	Solution Applicability	Solution Description	Solution Efficacy
	When nlist() is used in a privileged program.	Operate at a level of privilege appropriate to the user so that the system will guard against inappropriate access to a substituted file.	Effective when feasible.
	Generally applicable.	The most basic advice for TOCTOU vulnerabilities is to not perform a check before the use. This does not resolve the underlying issue of the execution of a function on a resource whose state and identity cannot be assured, but it does help to limit the false sense of security given by the check.	Does not resolve the underlying vulnerability but limits the false sense of security given by the check.
	Generally applicable.	Limit the interleaving of operations on files from multiple processes.	Does not eliminate the underlying vulnerability but can help make it more difficult to exploit.
	Generally applicable.	Limit the spread of time (cycles) between the	Does not eliminate the underlying vulnerability

		check and use of a resource.	but can help make it more difficult to exploit.
	Generally applicable.	Recheck the resource after the use call to verify that the action was taken appropriately.	Effective in some cases.
Signature Details		int nlist(const char *file_name, struct nlist *nl);	
Examples of Incorrect Code		<pre>// assume we are running as suid-root struct nlist nl[100]; // populate nl with names of symbols to be examined nlist("someExecutable", nl); // print symbol table information</pre>	
Examples of Corrected Code		<pre>// assume we are running as suid-root struct nlist nl[100]; // populate nl with names of symbols to be examined // change effective user and group IDs to reflect user nlist("someExecutable", nl); // print symbol table information</pre>	
Source References		<ul style="list-style-type: none"> • ITS4 Source Code Vulnerability Scanning Tool² • http://seclab.cs.ucdavis.edu/projects/vulnerabilities/scriv/ucd-ecs-95-09.pdf³ • http://www.phrack.org/show.php?p=60&a=6 	
Recommended Resources		<ul style="list-style-type: none"> • AIX man page for nlist()⁵ • HP-UX man page for nlist()⁶ 	
Discriminant Set		Operating System	<ul style="list-style-type: none"> • UNIX (All)
		Languages	<ul style="list-style-type: none"> • C • C++

Cigital, Inc. Copyright

Copyright © Cigital, Inc. 2005-2007. Cigital retains copyrights to this material.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

For information regarding external or commercial use of copyrighted materials owned by Cigital, including information about “Fair Use,” contact Cigital at copyright@cigital.com¹.

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

1. <mailto:copyright@cigital.com>